LU02a - Klassendiagramm und Klasse

In einem ersten Schritt beschränken wir uns auf die Darstellung einer Klasse in der Notationssprache UML (Unified Modelling Language).

- color: String
- door_is_open: boolean
- door_is_locked: boolean
- the_door_lock: Door_Lock

+ Door(ref2door_lock: Door_Lock, base_color: String)
+ open_the_door(): void
+ close_the_door(): void
+ lock_the_door(): void
+ unlock_the_door(): void
+ test(): void
+ door_is_open(): boolean
+ «prop» door_is_locked(): boolen
+ «grop» color(): Color
+ «setter» color(new_color: String): void

Klassenbezeichnung (Klassenname)

Attribute (Werte) des Objekts sowie Referenzen auf andere Objekte

Konstruktoren des Objekts und Methoden

Abb: Klassendigramm nach UML

https://de.wikipedia.org/wiki/Klassendiagramm

Die «Übersetzung» des Klassendiagramms in der Abbildung in den entsprechenden Code finden Sie weiter unten.

Erklärung zu den Elementen des Klassendiagramms:

- **Attribute** sollten immer private deklariert werden, um deren Wertebereich garantieren zu können. [*Die Erklärung zu data hiding folgt in Kapitel 5*]
- **Konstruktoren** werden bei der Erzeugung eines Objekts als erstes ausgeführt und dienen primär dem Initilaisieren der Attribute.
- **Methoden** stellen die Funktionalität der Klasse dar. Oft wird die Gesamtheit aller Methoden auch als «die Schnittstelle» der Klasse bezeichnet.

Im folgenden Code-Stück werden diese Elemente zusätzlich beschrieben.

Beispiel 1.5: Abstraktion einer Türe

```
class Door:
    """
    Diese Klasse beschreibt eine Türe mit der Eigenschaft color (Farbe) und
den Zuständen
    door_is_open (für geöffnete Türe) sowie door_is_locked (für verriegelte
Türe).
    Die Türe überwacht die beiden Zustände und verhindert so Aktionen, die
nicht möglich sind.
```

```
Das Verriegeln selber delegiert die Türe an ein Objekt vom Typ Door lock
(Türschloss).
    .....
# Mit dem Keyword def wird eine Funktion bzw. eben ein Konstruktor
deklariert.
# Der Konstruktor trägt IMMER den Namen init und weist als ersten
Parameter den Wert self auf.
# Danach folgen die Übergabeparameter, deren Werte dann den Attributen
zugewiesen werden.
# Attribute können aber auch mit einem fixen Wert initialisiert werden.
# Konstruktoren werden als Erstes im Programm angeschrieben.
    def init (self, ref2door lock, base color):
        Erzeugt ein Tür-Objekt.
        :param ref2door_lock:
        :param base color:
# ein privates Attribut muss im Konstruktor initialisiert werden und ist
dann in der Klasse
# über self. name des Attributs ansprechbar.
        self. the door lock = ref2door lock
# Hier wird der Setter eines Attributs aufgerufen (siehe unten)
        self.color
                              = base color
        self._door_is_open
                             = False
        self. door is locked = False
# Nach den Konstruktoren folgen Methoden, die eine Verarbeitung auslösen.
# Danach folgen Methoden, die auf ein Ereignis reagieren
    def open the door(self):
       Methode für das öffnen der Türe.
       Das ist aber nur möglich, wenn die Türe nicht verriegelt ist.
        if self._door_is_locked == False:
            self. door is open = True
   def close the door(self):
        0.00
        Methode für das schliessen der Türe.
       Das geht immer, auch wenn die Türe schon geschlossen oder verriegelt
ist. Der Zustand ändert dann nämlich nicht.
        self._door_is_open = False
   def lock the door(self):
        Methode für das verriegeln der Türe.
       Das ist nur möglich, wenn die Türe nicht offen ist.
        Für das verriegeln ist aber das Türschloss zuständig. Es weiss wie
```

https://wiki.bzz.ch/ Printed on 2025/11/20 04:42

```
das geht.
        if self. door is open == False:
            self. door is locked = self. the door lock.lock()
   def unlock_the_door(self):
        Methode für das entriegeln der Türe
        Das ist nur möglich, wenn die Türe verriegelt ist.
        Für das entriegeln ist aber das Türschloss zuständig. Es weiss wie
das geht.
        0.000
        if self. door is locked:
            self._door_is_locked = self._the_door_lock.unlock()
   def test(self):
        schreibt alle Attribute in den StdOut
        print(f'Türfarbe : {self.color}')
        print(f'Türe offen: {self. door is open}')
        print(f'Türe verriegelt: {self._door_is_locked}')
# Am Ende folgen die getter- und setter-Methoden für die Attribute der
Klasse
# getter werden mit der Anotation @property markiert.
   @property
   def door is open(self):
        getter-Methode für den Zustand door is open
        :return: true, wenn die Türe offen ist, sonst false
        return self. door is open
   @property
    def door_ist_locked(self):
        getter-Methode für den Zustand door_is_locked
        :return: true, wenn die Türe verriegelt ist, sonst false
        return self._door_is_locked
   @property
   def color(self):
        getter-Methode für die Eigenschaft color
        :return: die Farbe des Objekts
        return self. color
# setter werden mit der Anotation @name.setter markiert.
```

```
@color.setter
    def color(self, new_color):
        setter-Methode für die Eigenschaft color
        :param new_color:
        self. color = new color
0.00
nur für die korrekte Übersetzung und Ausführung
class DoorLock:
    dummy Klasse, damit in der Klasse Tuere kein Fehler auftritt
    def init__(self):
        print("ein Schloss erzeugt")
    def lock(self):
        return True
    def unlock(self):
        return False
# Hier die main-Methode festlegen
if name == ' main ':
    print('Test für Tür-Objekt')
    the door lock = DoorLock()
               = Door(the door lock, 'grün')
    the door
    the door.test()
    print('-- Türe jetzt öffnen')
    the_door.open_the_door()
    the door.test()
```



```
https://wiki.bzz.ch/ - BZZ - Modulwiki
```

Permanent link:

https://wiki.bzz.ch/modul/m320_2024/learningunits/lu02/klassendiagramm?rev=1713370502

Last update: 2024/04/17 18:15



https://wiki.bzz.ch/ Printed on 2025/11/20 04:42